Training a Sigmoidal Node is Hard

Don R. Hush

CIC-3, MS B265
Los Alamos National Laboratory
Los Alamos, NM 87545

August 1, 1998

Abstract

This paper proves that the task of computing near–optimal weights for sigmoidal nodes under the L_1 regression norm is NP–Hard. For the special case where the sigmoid is piecewise–linear we prove a slightly stronger result, namely that computing the *optimal* weights is NP–Hard. These results parallel that for the one–node pattern recognition problem, namely that determining the optimal weights for a threshold logic node is also intractable. Our results have important consequences for constructive algorithms that build a regression model one node at a time. It suggests that although such methods are (in principle) capable of producing efficient size representations (e.g. see Barron (1993); Jones (1992)), finding such representations may be computationally intractable. These results holds only in the deterministic sense, that is they does not exclude the possibility that such representations may be found efficiently with high probability. In fact it motivates the use of heuristic and/or randomized algorithms for this problem.

Keywords: nonlinear regression, sigmoids, piecewise-linear regression, multilayer networks, computational complexity

1 Introduction

This paper is concerned with the computational complexity of the training problem for neural networks whose hidden layer nodes perform the familiar affine projection of the input followed by a nonlinear activation function, that is

$$y = \sigma(w_0 + \sum_{i=1}^{d} w_i x_i)$$

where $\{x_i\}$ are the node inputs, $\{w_i\}$ are the node parameters (or weights), and $\sigma(\cdot)$ is the node activation function (typically sigmoidal). Experience has shown that the training process for such networks can be computationally expensive, especially for larger problems with high dimensional inputs and/or large data sets. It remains an open problem to characterize the intrinsic complexity of the training problem in its full generality, but numerous restricted results are available, most suggesting the intractability of the problem. These results depend heavily on the specifics of the problem definition, e.g. the network topology, the type of activation function(s), the characteristics of the training data, the training criterion, and the question being asked. Many of these results are developed under the framework of the *loading problem*, defined by Judd (1990).

The Loading Problem: Given a network specification (i.e. a description of the topology, the node activation functions, etc.) and a set of training samples $S = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$; does there exist a set of weights for which the network produces the correct output on all N training samples (i.e. can the data be "loaded" onto the network)? More precisely, if we let $f(\mathbf{x})$ denote the function performed by the network; does there exist a set of weights for which $f(\mathbf{x}_i) = y_i$ for all $i = 1, 2, ..., N$?

Note that the loading problem is posed as a *decision* problem. It does not ask for the production of weights, merely for a yes or no answer as to their existence. This is typical of decision problems which are abstracted from optimization problems. The most important difference between the loading problem and a typical situation encountered in practice however, is that it asks the question about a *zero error* solution. In practice we often expect the "best" solution to have nonzero error. Nevertheless, it would appear that answering the question for the zero error case is no harder than for the minimum error case, and so it can be argued that the complexity of loading is important.

Most results for the loading problem assume a threshold logic node (i.e. a node with hard–limiting activation function) at the output. This makes the optimization problem combinatorial. The target outputs, y_i, are thus taken from $\{0, 1\}$ so that zero error is possible. It is also convenient to work with binary input data, i.e. $\mathbf{x} \in \{0, 1\}^d$. Under this setting the following results are characteristic of those produced to date. Judd showed that the loading problem is NP–Complete for a class of sparsely connected threshold logic networks with a special (nonconventional) topology Judd (1990). Blum and Rivest (1988) proved that loading is NP–Complete for what many consider to be the simplest possible one hidden layer network, a 3–node threshold logic network with two nodes in the hidden layer. Höffgen (1993) proved that loading a 3–node network with continuous sigmoid activations is NP–Hard under the restriction of binary weights. DasGupta *et al.* (1995) proved that loading a 3–node network with piecewise–linear activations in the hidden layer is NP–Complete. Šíma (1996) proved that loading a 3–node network with continuous sigmoidal activations in the hidden layer is NP–Hard under the constraint that the output bias weight is zero. Finally, Lin and Vitter (1991) proved that loading is NP–Complete for the smallest possible threshold logic network with cascade architecture (i.e. a 2–cascade network).

Not all complexity results have been developed under the loading framework. For example, using a refinement of the PAC–learning framework, Maass (1995) describes architectures for which efficient learning algorithms do exist. Maass's networks map real–valued inputs to real–valued outputs and use piecewise–polynomial activation functions. These activation functions differ from the piecewise–linear activations considered later in this paper (and in DasGupta *et al.* (1995); Jones (1997)) in that they are discontinuous and may have many piecewise components.

In addition, Jones (1997) has shown that training a 3–node network with sigmoidal hidden layer nodes and a linear output node is NP–Hard. His proof requires that the sigmoidal activation functions in the hidden layer satisfy certain monotone and Lipschitz conditions. The network maps real–valued inputs to real–valued outputs, and the NP–Hardness result applies under two different criterion: the L_2 error norm and the *minimax* error norm. Jones also shows that the problem is NP–Complete under the additional assumptions that either σ is piecewise–linear and the L_2 error is used or, σ is piecewise–rational and the *minimax* error norm is used.

The results above suggest that the training problem for most neural network models is intractable. Baum (1989) points out however that the intractability may be due (in part) to the fact that the network architectures are fixed during training. He suggests that the learning problem may be intrinsically easier if we are allowed to add nodes and/or weights during the process. Algorithms of this type are typically called *constructive algorithms*. In most cases the constructive approach cannot guarantee that the resulting network is of minimal size, but in many cases we can expect the size to be reasonable (more on this below). The hope then is that learning can be accomplished more efficiently if it is performed one node at a time. The tractability of this approach hinges on the complexity of training for a single node.

In this context it is interesting to note that the loading problem can be solved in polynomial–time for a threshold logic node (e.g. using linear programming). However, if the answer to the loading problem is "no" (i.e. zero error is not achievable), then the problem of finding the optimal node (i.e. the one with fewest errors) is computationally intractable (e.g. see Siu *et al.* (1995)). A careful study of this problem reveals that the intractability stems from the dimensionality of the input. That is, if the input dimension is fixed then the problem admits a polynomial–time solution (although the degree of the polynomial scales with the dimension, so it may not be practical for problems of even modest dimension).

There are few results concerning the complexity of learning for a single node with real–valued output (e.g. a regression node). If the node is linear then it typically admits an efficient solution, either in the form of an algorithm for systems of linear equations, or in the form of a Linear Programming problem. But little is known about the complexity when the node is nonlinear. This is the issue addressed here. We prove a complexity result for the popular class of sigmoidal nonlinearities.

To address the issue of size mentioned above, and to further motivate the results in this paper, we consider the work of Barron (1991, 1993) and Jones (1992). Their results pertain to one–hidden layer networks with sigmoidal activations in the hidden layer and a linear output. Barron (1991, 1993) has shown that when the function being modeled by the network belongs to a particular class of continuous functions, Γ_C, the generalization error for the network (under the expected L_2 norm) is bounded by

$$O\left(\frac{1}{n}\right) + O\left(\frac{nd\log N}{N}\right)$$

where n is the number of hidden layer nodes, d is the dimension of the input, and N is the number of training samples. The first term, $O(1/n)$, is the *approximation* error and is due to the inability of a finite–size network to produce a zero–error model for functions in Γ_C. The second term, $O(nd\log N/N)$, is the *estimation* error and is due to the fact that the model must be inferred from only a finite number of data samples. Of particular interest is the $O(1/n)$ bound on approximation error, which is a significant improvement over the $O(1/n^{1/d})$ form achieved with fixed basis function models Barron (1993). It has been shown that this $O(1/n)$ bound can be achieved *constructively*, that is by designing the nodes one at a time Barron (1993); Jones (1992). The proof of this result is itself constructive, and thus provides a framework for the development of an algorithm which can (in principle) achieve this bound. It starts by fitting the first node to the original function. The second node is then fit to the residual from the first approximation, and the two are combined to form the second approximation. This process of fitting a node to the current residual and then combining it with the previous approximation continues until a suitable size model is found. The proof that this algorithm can achieve an $O(1/n)$ rate of approximation relies on the assumption that the node produced at each step is within $O(1/n^2)$ of the optimum Barron (1993); Jones (1992). In practice a node may fall short of the optimum due to the error introduced by a finite training set. However, using the estimation error result for $n = 1$, if the number of training samples satisfies $N/\log N = \Omega(n^2 d)$ then, with perfect learning, the error will (on average) satisfy the $O(1/n^2)$ tolerance making the $O(1/n)$ rate achievable. Perfect learning implies that the training procedure is able to produce the optimal set of weights. Thus, if this training problem can be solved efficiently, then we have conditions under which Barron's approximation rate could be realized in practice. In this setting however, the efficiency of training remains an open problem Barron (1993). This paper takes a step towards addressing this problem by answering this question in the negative for sigmoidal nodes that are trained using the L_1 norm. Note that the Jones/Barron results hold under the L_2 norm, while our hardness result uses the L_1 norm. It is not clear that the Jones/Barron results can be extended to the L_1 norm, although qualitatively similar results are likely. On the other hand, even though we have not yet discovered a hardness proof using the L_2 norm, the results in this paper suggest very strongly that such a proof exists.

2 Problem Statement and Main Result

The precise computational problem that we wish to address is defined as follows.

Approximately Optimal Sigmoid (APP-OPT-σ): *Let $\sigma : \Re \to [0,1]$ be an activation function. Given a regression data set $S = \{(\mathbf{x}_i, y_i)\}$ with N pairs $(\mathbf{x}_i, y_i) \in \Re^d \times \Re$, does there exist a set of parameters $\mathbf{w} \in \Re^d$, $w_0 \in \Re$ so that*

$$E_1 = \sum_{i=1}^{N} |y_i - \sigma(\mathbf{w}^T \mathbf{x}_i + w_0)|$$

is strictly within 1 of its infimum?

The following theorem gives the main result.

Theorem 1 *For any non–decreasing function σ, APP-OPT-σ is NP–Hard.*

Note that it is essential to require only approximate optimality since an infimum may not be achievable with finite weights. This is true for example with the smooth sigmoid that is commonly used in neural network models. This characteristic is not true of all σ however, and in such cases it may be possible to provide an even stronger result. In section 3 we show that when σ is piecewise–linear, a similar hardness result is achievable under the condition of exact optimality.

The proof of Theorem 1 uses a reduction from the *Maximum Linearly Separable Subset* (MLSS) problem which we now describe. Let us define a *pattern recognition data set* to be a set of labeled binary patterns $P = \{(\mathbf{x}_i, \alpha_i)\}$ such that $\mathbf{x}_i \in \{0,1\}^d$ are the pattern vectors and $\alpha_i \in \{-,+\}$ are the labels. Let $P_+ = \{\mathbf{x}_i : \alpha_i = +\}$ and $P_- = \{\mathbf{x}_i : \alpha_i = -\}$ be the subsets of patterns from the two pattern classes. A linear dichotomy of P is a partitioning of the pattern vectors into two (disjoint) subsets according to

$$\begin{aligned} L_+ &= \{\mathbf{x}_i : \mathbf{a}_l^T \mathbf{x}_i \geq a_0\} \\ L_- &= \{\mathbf{x}_i : \mathbf{a}_l^T \mathbf{x}_i < a_0\} \end{aligned} \tag{1}$$

This dichotomy is characterized by the $(d+1)$–dimensional parameter vector $\mathbf{a}^T = [a_0, \mathbf{a}_l^T] = [a_0, a_1, ..., a_d] \in \Re^{d+1}$. A set P is said to be *linearly separable* if and only if there exists a parameter vector $\mathbf{a}^*$ such that $L_+ = P_+$ and $L_- = P_-$. Determining the linear separability of P can be accomplished in polynomial–time (e.g. using linear programming). If P is not linearly separable however, determining the best linear partitioning is a computationally intractable problem. For example, let us define a *maximum linearly separable subset* of P to be a subset $P' \subseteq P$ that is linearly separable and has maximum cardinality. The decision version of this problem, stated below, is NP–Complete, e.g. see Siu *et al.* (1995).

Maximum Linearly Separable Subset (MLSS): *Given a pattern recognition data set P,* *and a positive integer $K \leq |P|$, does there exist a linearly separable subset $P' \subseteq P$ with $|P'| \geq K$?*

Now, the proof of Theorem 1 is accomplished by providing a polynomial–time reduction from MLSS to APP-OPT-σ. We use a Turing reduction that makes a single call to an oracle for APP-OPT-σ. The reduction is comprised of the three steps shown below.

Reduction: MLSS $\leq_P$ APP-OPT-σ.

1. Transform the pattern recognition data set P into a regression data set S by mapping each pattern vector in P directly to a regression vector in S, and each label in P to a response variable in S according to

$$y_i = \begin{cases} 1, & \alpha_i = + \\ 0, & \alpha_i = - \end{cases}$$

2. Call the oracle for APP-OPT-σ with S as the input. The oracle returns (near)–optimal parameters $\mathbf{w}, w_0$.

3. Use the parameters from Step 2 to compute the set of projected samples

$$Z = \{z_i : z_i = \mathbf{w}^T \mathbf{x}_i + w_0\}$$

   Position $m \leq N$ step functions to pass between the $m \leq N$ distinct values of z_i and let $\{Z_+, Z_-\}_j$ be the partition induced by the j^{th} step function. Let M_j be the number of patterns that are correctly labeled by the j^{th} step function, and set $M^* = \max_j M_j$. If $M^* \geq K$ then answer "yes", otherwise answer "no".

Proof of Correctness for Reduction:
First note that the reduction can be carried out in polynomial–time (it is linear in the size of the input). The heart of the proof is in showing that when presented with pattern recognition data, the oracle for APP-OPT-σ will return a solution from which the maximum linearly separable subset can be extracted as in step 3. This relies on two simple observations about the relationship between step functions and any non–decreasing, bounded σ.

The first observation is that σ is equivalent to a convex combination of step functions on any finite set. That is, for any finite set $Z \subset \Re$ there exist step functions $s_1, ..., s_m$ and convex coefficients $\alpha_1, ..., \alpha_m$ such that

$$\sigma(z) = \sum_{i=1}^{m} \alpha_i s_i(z) \tag{2}$$

for all z in Z. Furthermore, $m \leq |Z|$ will suffice, since we can choose the steps to occur to the left of the smallest point and between the points.

The second observation is that σ can approximate a step function. That is, for any finite set $Z \subseteq \Re$, for all $\delta > 0$, and for all step functions s, there is an $a \in \Re$ for which $|\sigma(az) - s(z)| < \delta$ for all z in Z. Thus, if the regression problem presented to σ requires a step function as its optimal solution (as it does in the reduction above), then σ can approximate that solution arbitrarily closely.

The following lemma formalizes these observations and provides the essential link needed to complete our proof.

Lemma 1 *Given any APP-OPT-σ problem for which $y_1, ..., y_N \in \{0, 1\}$, and any $\mathbf{w}, w_0$, there is a step function s and real constants a, b such that*

$$\sum_{i-1}^{N} |y_i - s(a\mathbf{w}^T\mathbf{x}_i + b)| \leq \sum_{i-1}^{N} |y_i - \sigma(w^T\mathbf{x}_i + w_0)|$$

Proof: To prove this we use (2) to write

$$\begin{aligned} \textstyle\sum_{i=1}^{N} |y_i - \sigma(\mathbf{w}^T\mathbf{x}_i + w_0)| &= \textstyle\sum_{i=1}^{N} |y_i - \sum_{j=1}^{m} \alpha_j s_j(\mathbf{w}^T\mathbf{x}_i + w_0)| \\ &= \textstyle\sum_{i=1}^{N} \Big(y_i - (2y_i - 1)\sum_{j=1}^{m} \alpha_j s_j(\mathbf{w}^T\mathbf{x}_i + w_0)\Big) \\ &= \textstyle\sum_{i=1}^{N} y_i + \sum_{j=1}^{m} \alpha_j \sum_{i=1}^{N}(1 - 2y_i)s_j(\mathbf{w}^T\mathbf{x}_i + w_0) \\ &= \textstyle\sum_{i=1}^{N} y_i + \sum_{j=1}^{m} \alpha_j J_j \end{aligned} \tag{3}$$

where

$$J_j = \sum_{i=1}^{N}(1 - 2y_i)s_j(\mathbf{w}^T\mathbf{x}_i + w_0) \tag{4}$$

Let $j^* = \arg\min_j J_j$. Clearly, since the α_j's are convex coefficients, (3) can be minimized by setting $\alpha_{j^*} = 1$ and the other coefficients to zero. This gives

$$\sum_{i=1}^{N} |y_i - \sigma(\mathbf{w}^T\mathbf{x}_i + w_0)| \geq \sum_{i=1}^{N} |y_i - s_{j^*}(\mathbf{w}^T\mathbf{x}_i + w_0)|$$

which proves the lemma. **QED**

Note that if we set σ equal to a step function s, then E_1 is an integer whose value is equal to the number of samples for which s disagrees with the training set. Thus, for step functions E_1 can change only by integer values.

Now consider the model returned by the oracle in Step 2 of the reduction above. The corresponding value of E_1 is within 1 of the infimum and by Lemma 1, the corresponding step function chosen in Step 3 has error within 1 of its infimum. Since the step function error can change only by an integer, the step function produced in Step 3 must be optimal. That is, the value of E_1 for this step function is infimal over all functions. Thus, the partition induced in Step 3 minimizes disagreements with the training set and therefore maximizes M_{j^*}. This completes the proof of correctness for the reduction. **QED**

It is worth noting that the above result can be easily extended to σ with any other bounded range by simply changing the labels y_i in the reduction.

3 Piecewise–linear Sigmoids

In this section we consider the special case where σ is piecewise–linear. Although this activation is less popular than the smooth sigmoid, its piecewise nature can be exploited to develop more efficient heuristic algorithms for learning, e.g. see Breiman and Friedman (1994); Hush and Horne (1998); Staley (1995). It is also simpler to evaluate in that it requires only addition, multiplication and comparison operations, in contrast to the trigometric function that must be evaluated for the smooth sigmoid. In addition, it provides a unique connection between sigmoid functions and linear splines, which are also commonly used in regression (a piecewise–linear sigmoid can be formed from two linear splines).

The piecewise–linear sigmoidal (PWLS) performs a mapping $\sigma : \Re^d \rightarrow \Re$ according to

$$\sigma(\mathbf{x}) = \begin{cases} w_+, & \mathbf{w}_l^T\mathbf{x} + w_0 \geq w_+ \\ \mathbf{w}_l^T\mathbf{x} + w_0, & w_- < \mathbf{w}_l^T\mathbf{x} + w_0 < w_+ \\ w_-, & \mathbf{w}_l^T\mathbf{x} + w_0 \leq w_- \end{cases} \tag{5}$$

where $\mathbf{x}^T = [x_1, x_2, ...x_d] \in \Re^d$ is the input vector, and

$$\mathbf{w}^T = [w_0, \mathbf{w}_l^T, w_+, w_-] = [w_0, w_1, ..., w_d, w_+, w_-] \in \Re^{d+3}$$

is the parameter vector that characterizes the node function [1]. An example of the surface formed by a PWLS node for a two–dimensional input is shown in Figure 1. It is comprised of three hyperplanes joined pairwise continuously at two hinge locations. These hinges induce linear partitions on the input space that divide the space into three regions as shown. Note that the parameterization in (5) allows complete flexibility in the placement of the PWLS surface in $\Re^d$, i.e. it can have arbitrary orientation, scale and offset.

The definition above differs from other common definitions in that the flat portions are not restricted to the values 0 and 1. In a multilayer network (with linear output) where all nodes are trained simultaneously, the 0/1 restrictions do not present a limitation, since the node outputs are scaled and shifted by weights in the subsequent layer. But when the hidden layer nodes are trained individually to match a target with arbitrary scale and offset (as they are in constructive learning algorithms), the 0/1 restrictions are not appropriate, hence the definition above.

Note that the PWLS node partitions the input vectors into three subsets S_+, S_l and S_- given by

$$\begin{aligned} S_+ &= \left\{\mathbf{x}_i : \mathbf{w}_l^T\mathbf{x}_i + w_0 \geq w_+\right\} \\ S_l &= \left\{\mathbf{x}_i : w_- \leq \mathbf{w}_l^T\mathbf{x}_i + w_0 \leq w_+\right\} \\ S_- &= \left\{\mathbf{x}_i : \mathbf{w}_l^T\mathbf{x}_i + w_0 \leq w_-\right\} \end{aligned} \tag{6}$$

In Hush and Horne (1998) we show that there are $\Theta(N^{d+1})$ such partitions. Further, with the partition fixed, the problem of learning the optimal weights can be cast as either a Linear

[1] Note that by definition $w_- < w_+$ (i.e. they cannot be equal).

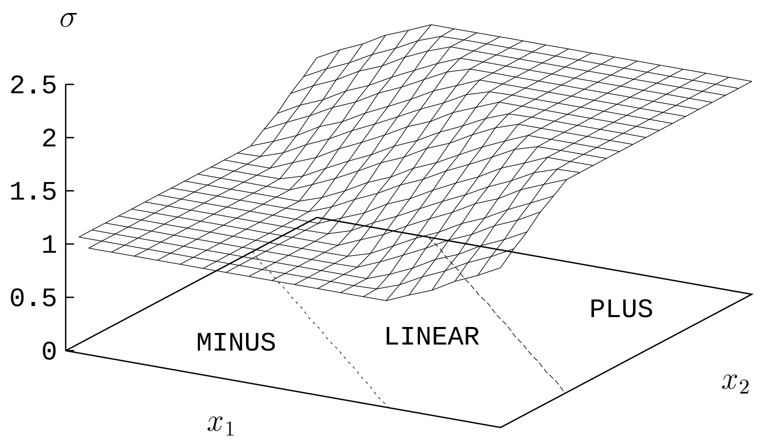

Figure 1: A piecewise–linear sigmoidal function in two dimensions.

Programming problem under the L_1 norm, or a Quadratic Programming under the L_2 norm. This suggests that the learning problem for this type of node is inherently combinatorial. It is also apparent that the complexity of learning stems from the exponential number of partitions (since, under reasonable assumptions, both LP and QP can be solved in polynomial–time).

The training problem considered here differs from that in the previous section in that we ask for an *optimal* rather than *near–optimal* solution.

Optimal Piecewise–linear Sigmoid (OPWLS): *Given a regression data set* $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ *and a PWLS node defined in (5), determine the parameter vector* $\mathbf{w}^*$ *that minimizes* E_1*, i.e.*

$$\mathbf{w}^* = \arg \min_{\mathbf{w} \in \Re^{d+3}} E_1$$

Theorem 2 *The OPWLS problem is NP–Hard.*

Proof: The proof of this theorem is the same as for Theorem 1, except that Step 2 of the reduction calls the oracle for OPWLS which, by definition, returns weights that minimize E_1. In this case the PWLS node will achieve the same minimal value of E_1 as the step function selected in Step 3. That is, since there is always a measurable interval between distinct samples, the PWLS node can dichotomize the data optimally using finite weights (e.g. by partitioning samples into S_+ and S_- and leaving S_l empty), and this solution achieves the lower bound on E_1 set by the step function. **QED**

4 Summary and Conclusion

This paper has shown that determining the weights that optimize the L_1 regression error for a sigmoid node is NP–Hard. Consequently it suggests that the problem of producing efficient size representations with constructive algorithms that build a regression model one node at a time may be computationally intractable. This result applies only in the deterministic sense, that is it does not exclude the possibility of finding efficient size representations in polynomial–time with high probability. In fact, our result motivates the use of heuristic and/or randomized algorithms for this problem.

The results in this paper suggest that a similar hardness result may exist for the same model under the L_2 regression norm, although the proof is likely to be quite different from the one presented here. A heuristic algorithm for solving the OPWLS problem under the L_2 regression norm can be found in Hush and Horne (1998).

Acknowledgments: I would like to acknowledge the BSP group at UNM and the Universidad de Vigo for their continued support and fruitful discussions of this work. I would also like to thank Fernando Lozano for his careful review and critique of this manuscript. A special thanks goes out the referees who provided key insights into simplifications of the proofs, and also help expand their scope considerably.

References

Barron, A. (1991). Approximation and estimation bounds for artificial neural networks. In L. Valiant and M. Warmuth, editors, *Proceedings of the 4th Annual Workshop on Computational Learning Theory*, pages 243–249.

Barron, A. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, **39**(3), 930–945.

Baum, E. (1989). A proposal for more powerful learning algorithms. *Neural Computation*, **1**(2), 201–207.

Blum, A. and Rivest, R. (1988). Training a 3–node neural network is NP–complete. In *Proceedings of the Computational Learning Theory (COLT) Conference*, pages 9–18. Morgan Kaufmann.

Breiman, L. and Friedman, J. (1994). Function approximation using ramps. In *Snowbird Workshop on Machines that Learn.*

DasGupta, B., Siegelmann, H., and Sontag, E. (1995). On the complexity of training neural networks with continuous activation functions. *IEEE Transactions on Neural Networks*, **6**(6), 1490–1504.

Höffgen, K.-U. (1993). Computational limitations on training sigmoidal neural networks. *Information Processing Letters*, **46**, 269–274.

Hush, D. and Horne, B. (1998). Efficient algorithms for function approximation with piecewise linear sigmoidal networks. *to appear in IEEE Transactions on Neural Networks*.

Jones, L. (1992). A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training. *The Annals of Statistics*, **20**, 608–613.

Jones, L. (1997). The computational intractability of training sigmoidal neural networks. *IEEE Transactions on Information Theory*, **43**(1), 167–173.

Judd, J. (1990). *Neural Network Design and the Complexity of Learning*. MIT Press, Cambridge, MA.

Lin, J.-H. and Vitter, J. S. (1991). Complexity results on learning by neural networks. *Machine Learning*, **6**, 211–230.

Maass, W. (1995). Agnostic pac–learning of functions on analog neural nets. Technical Report NC–TR–95–002, NeuroCOLT Technical Report Series.

Siu, K.-Y., Roychowdhury, V., and Kailath, T. (1995). *Discrete Neural Computation: A Theoretical Foundation*. Prentice–Hall, Englewood Cliffs, NJ.

Staley, M. (1995). Learning with piece–wise linear networks. *International Journal of Neural Systems*, **6**(1), 43–59.

Šíma, J. (1996). Back–propagation is not efficient. *Neural Networks*, **9**(6), 1017–1023.